

PRZEMYSŁAW ŻARNECKI

PROGRAMOWANIE PYTHON 01

KILKA SPRAW ORGANIZACYJNYCH

- ▶ CEL ZAJĘĆ - zapoznanie się z podstawami języka. Krótkoterminowo - abyście mogli jakieś proste aplikacje na koniec móc samodzielnie tworzyć. Długoterminowo - aby można było później jakieś nieco bardziej zaawansowane rzeczy w Pythonie stworzyć.
- ▶ Podstawą zaliczenia będzie lista zadań - zrobiona w formie testu.
- ▶ Na zaliczenie ćwiczeń forma projektowa. Zazwyczaj dwie do wyboru. Zestaw mniejszych zadań + jakiś większy problem projektowy do wyboru

KILKA SPRAW ORGANIZACYJNYCH

- ▶ Widzimy się z tego co widzę 2x w tygodniu + wykład
- ▶ W razie potrzeby proponuję konsultacje po wykładzie, czy ćwiczeniach. Jak trzeba - to można się w innych terminach umawiać. Można zdalnie, na żywo.
- ▶ mail. przemyslaw.zarnecki@wroclaw.merito.pl. Awaryjnie - pzarnecki@protonmail.com - ale wszelkie sprawy formalne tylko przez oficjalnego
- ▶ Można kontakt przez teams. Ale również bywa. Proponuję szyfrowany komunikator element - @pzarnecki:one.ems.host.

ZANIM PRZEJDZIEMY DALEJ

- ▶ Zaczniemy od omówienia narzędzi pracy - z tym później wiążą się różnego rodzaju problemy
- ▶ Chociaż uważam, że na uczelniach nad tym czasem za bardzo się skupiamy - w pracy możecie dostać inne całkiem
- ▶ Mi zależy bardziej na umiejętnościach programowania
- ▶ Ale kilka informacji być musi

NAJPOPULARNIEJSZE IDE DO PYTHONA

The screenshot shows the PyCharm IDE interface. The top navigation bar includes 'wykłady' and 'Version control'. The left sidebar shows a 'Project' tree with 'wykłady' selected, containing 'venv', 'main.py', and 'External Libraries'. The main editor window displays 'main.py' with the following code:

```
# This is a sample Python script.  
# Press ^R to execute it or replace it with your code.  
# Press Double ↑ to search everywhere for classes, files, tool windows, actions, and settings.  
  
1 usage  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17
```

The code includes comments explaining how to run and debug the script. Line 16 contains a syntax error: '#See PyCharm help at <https://www.jetbrains.com/help/pycharm/>'. The bottom panel shows the 'Run' tab with a terminal output window displaying a syntax error message:

```
/Users/przemyslawzarnecki/PycharmProjects/wykłady/venv/bin/python /Users/przemyslawzarnecki/PycharmProjects/wykłady/main.py  
File "/Users/przemyslawzarnecki/PycharmProjects/wykłady/main.py", line 7  
    y = y + " World" #łączenie łańcuchów  
          ^  
SyntaxError: invalid character '' (U+201C)  
  
Process finished with exit code 1
```

The status bar at the bottom indicates the file is 'wykłady > main.py' and the Python version is 'Python 3.10 (wykłady)'.

PYCHARM

<https://www.jetbrains.com/pycharm/>

Macie możliwość za darmo wersji

Komercyjnej jako studenci ale musicie

jakby kupić tylko zaznaczyć, że jesteście

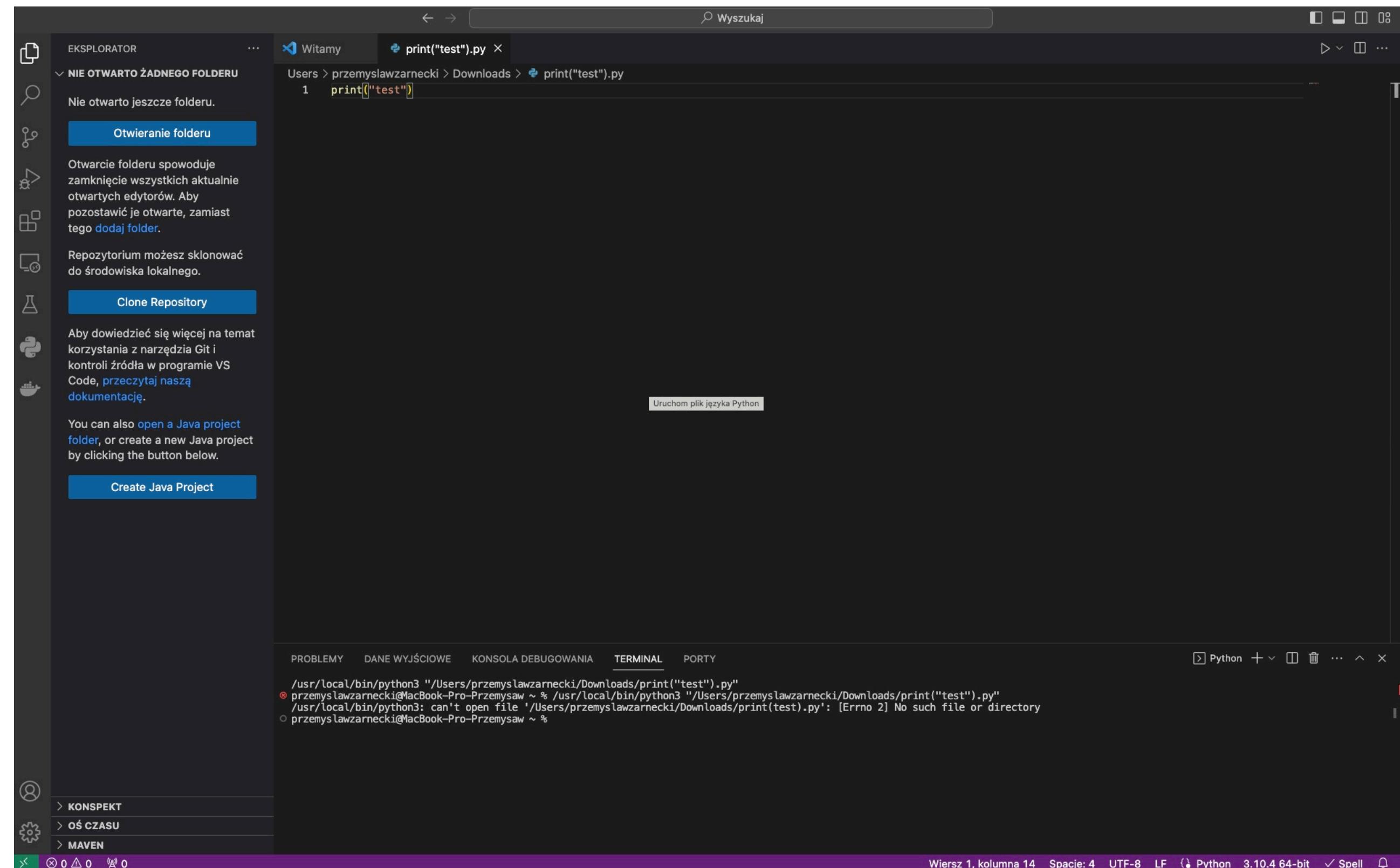
Studentami

PYCHARM - SPOSÓB NA ZDOBYCIE LICENCJI:

The screenshot shows the PyCharm website. At the top, there's a navigation bar with links: Developer Tools, Team Tools, Education (which is highlighted with a red box), Solutions, Support, Store, a search icon, and user account icons. Below the navigation is a banner for PyCharm, featuring a yellow and green abstract shape. The main content area has a section titled "Free License Programs" with two bullet points: one with a checkmark about renewing free for students/teachers, and one with a red X about not sharing. It also lists "Academic Licensing", "Open Source", "User Groups", "Events Partnership", and "Developer Recognition". A large illustration of a hand holding a smartphone with a colorful brain icon is on the left. In the center, text says "Get free access to all developer tools from JetBrains!" with a blue "Apply now" button. A red box highlights the "Apply now" button, and a red arrow points to it from below. At the bottom, there's a section titled "Options to get a free educational license" with icons for a house, a person, a laptop, and a book.

The screenshot shows the JetBrains Education page. The top navigation bar is identical to the PyCharm one. The main content area is divided into sections for "FOR LEARNERS", "FOR EDUCATORS", and "FOR TEAMS". Each section contains text and small icons. To the right, there's a large box titled "FREE LICENSES" with several sub-options: "For students and teachers", "For educational institutions", and "For bootcamps and courses". A red box highlights the "FREE LICENSES" title, and a red arrow points to it from the bottom left. At the bottom, there's a dark banner with text: "JetBrains tools are FREE for education! Enjoy a professional developer experience using industry-leading tools to learn, teach, and collaborate." and a "Get started now" button.

NAJPOPULARNIEJSZE IDE DO PYTHONA



VISUAL STUDIO CODE

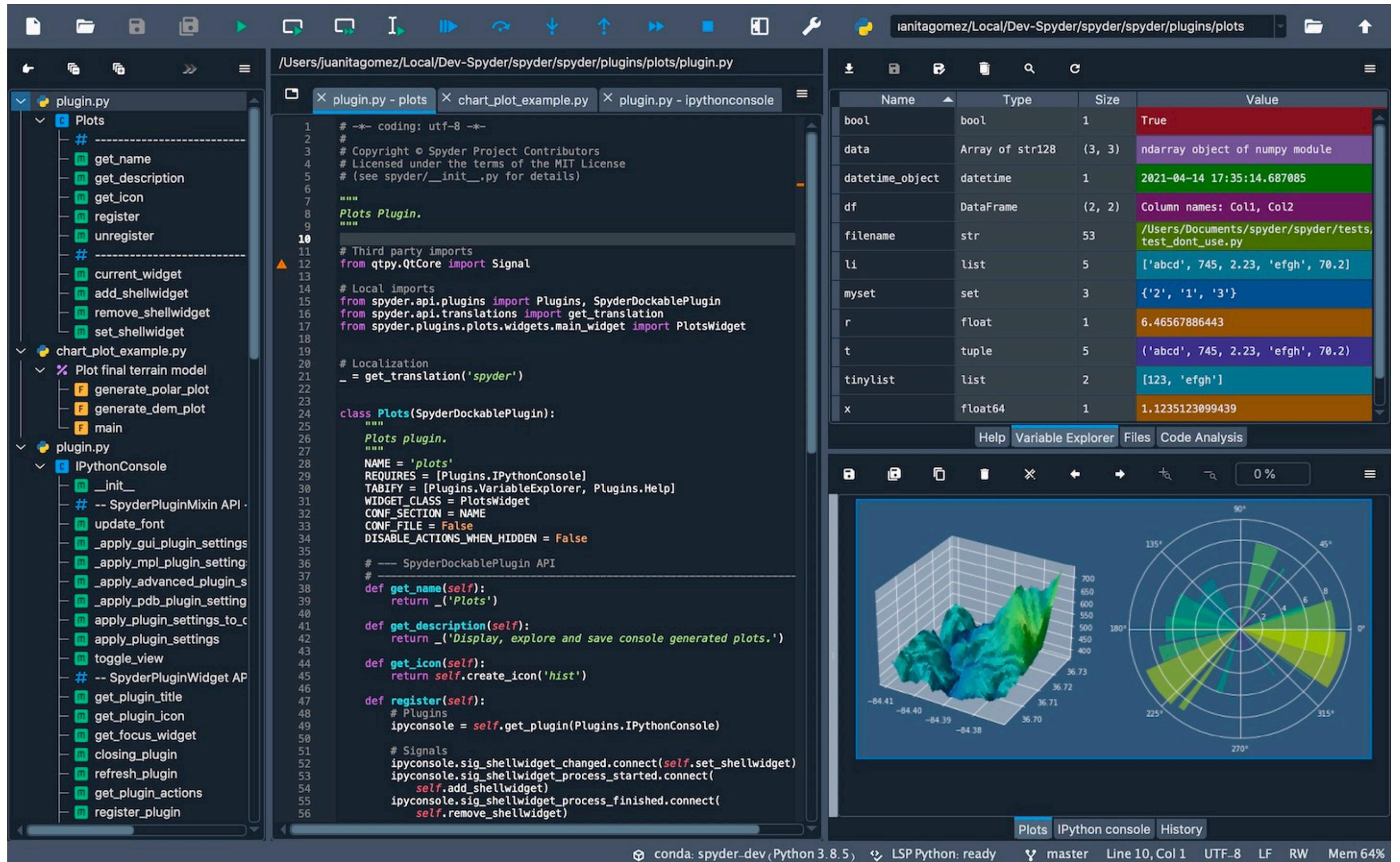
<https://code.visualstudio.com/>

Darmowa, OpenSource.

Jest ok. Ale jak się za dużo języków

Użyje, to potrafi nieco zamulić

NAJPOPULARNIEJSZE IDE DO PYTHONA



SPYDER

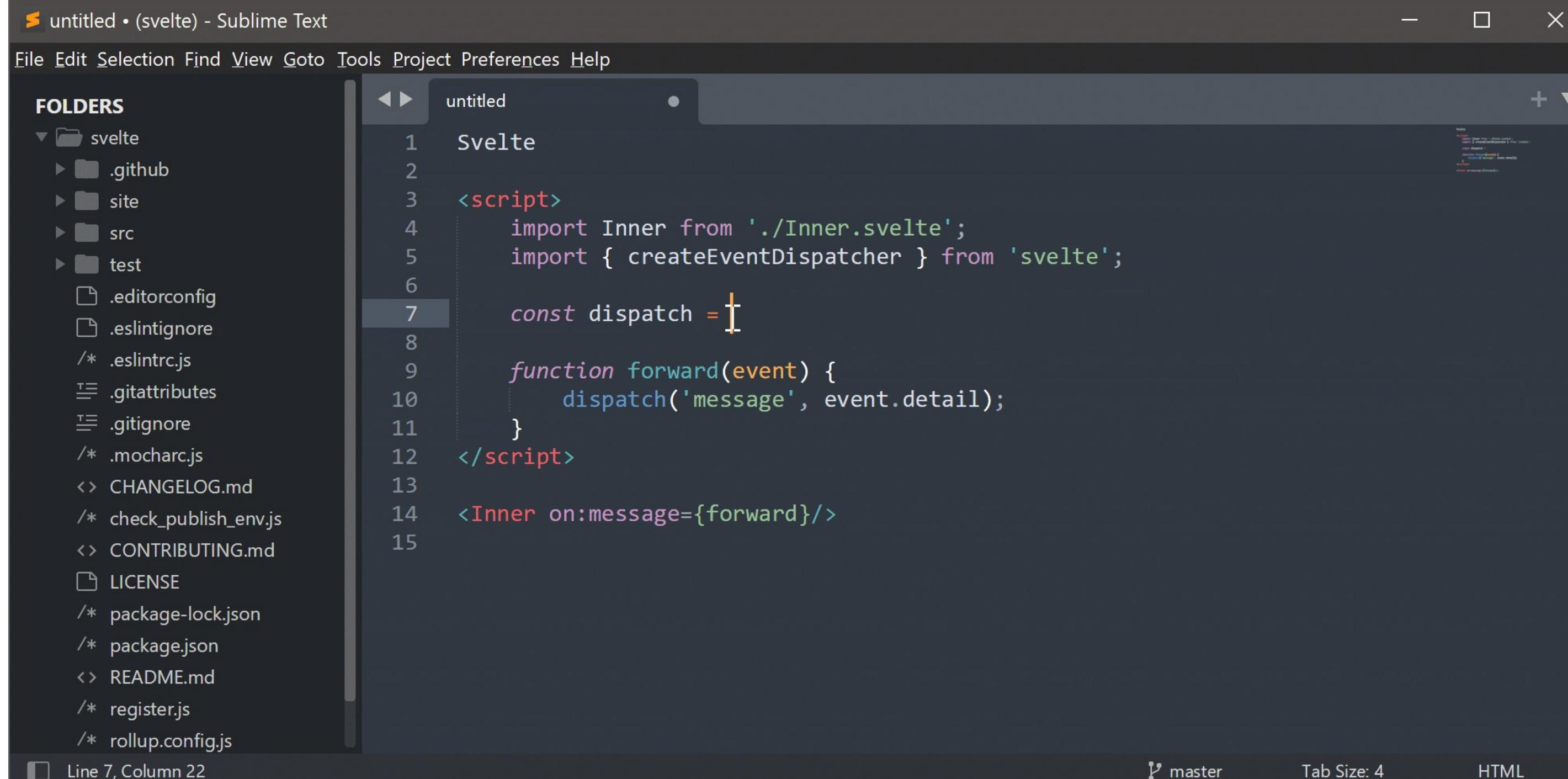
<https://www.spyder-ide.org/>

Dość ciekawa alternatywa

Dla PYCHARM. Ciut moim

Zdaniem szybciej działa

NAJPOPULARNIEJSZE IDE DO PYTHONA



The screenshot shows the Sublime Text 3 interface with a dark theme. The title bar says "untitled • (svelte) - Sublime Text". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. A "FOLDERS" sidebar on the left lists project files like svelte, .github, site, src, test, and various configuration and ignore files. The main editor window contains the following Python code:

```
1 Svelte
2
3 <script>
4     import Inner from './Inner.svelte';
5     import { createEventDispatcher } from 'svelte';
6
7     const dispatch = [
8         function forward(event) {
9             dispatch('message', event.detail);
10        }
11    }
12 </script>
13
14 <Inner on:message={forward}>
```

At the bottom of the editor, it says "Line 7, Column 22". The status bar at the bottom shows "master", "Tab Size: 4", and "HTML".

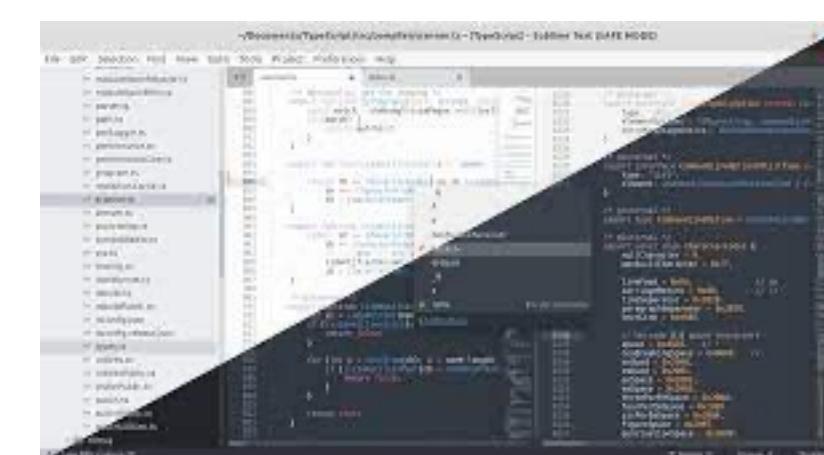
SUBLIME TEXT

<https://www.sublimetext.com/>

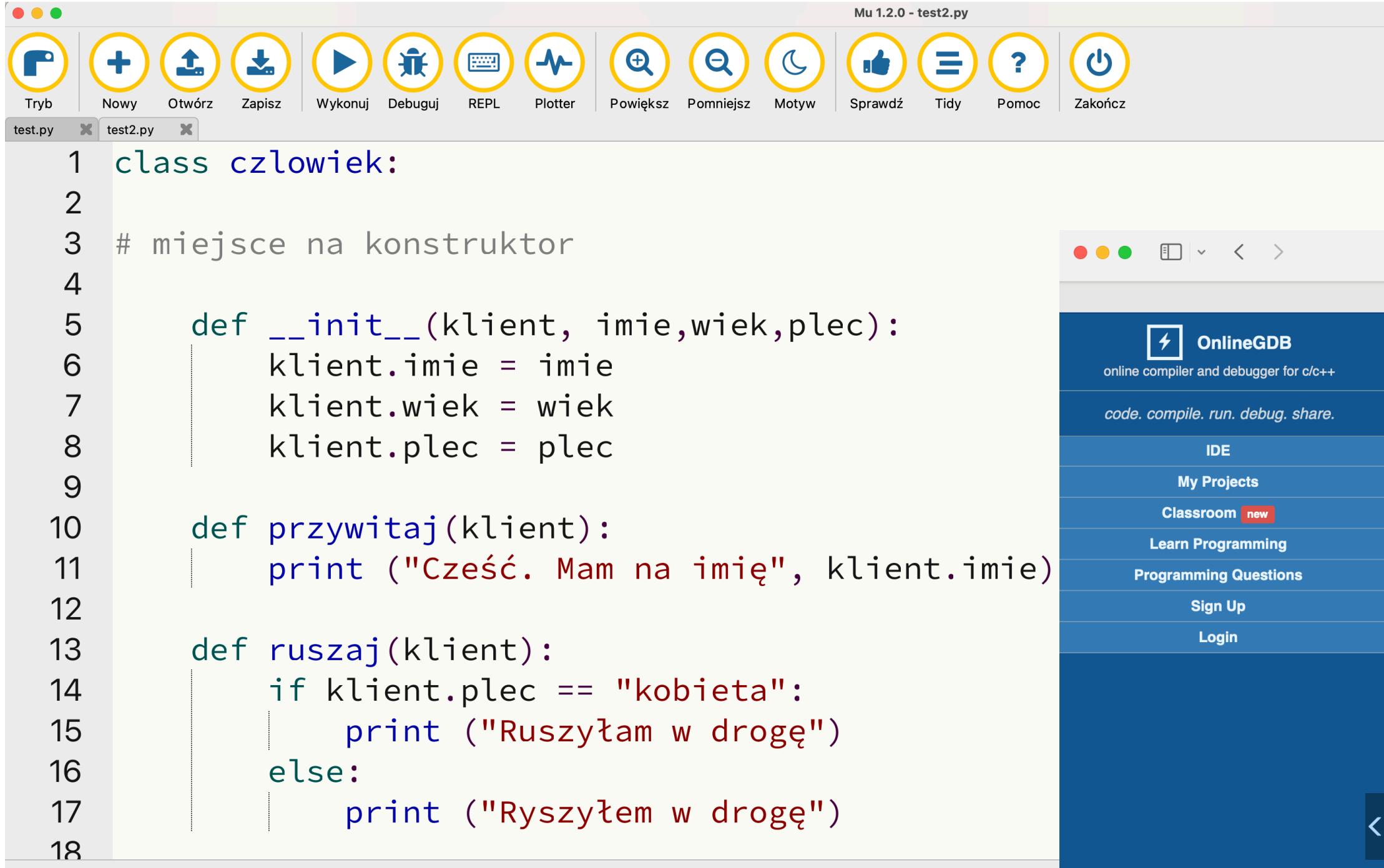
Taki nowoczesny edytor.

Edytor jest ogólnie szybszy od IDE.

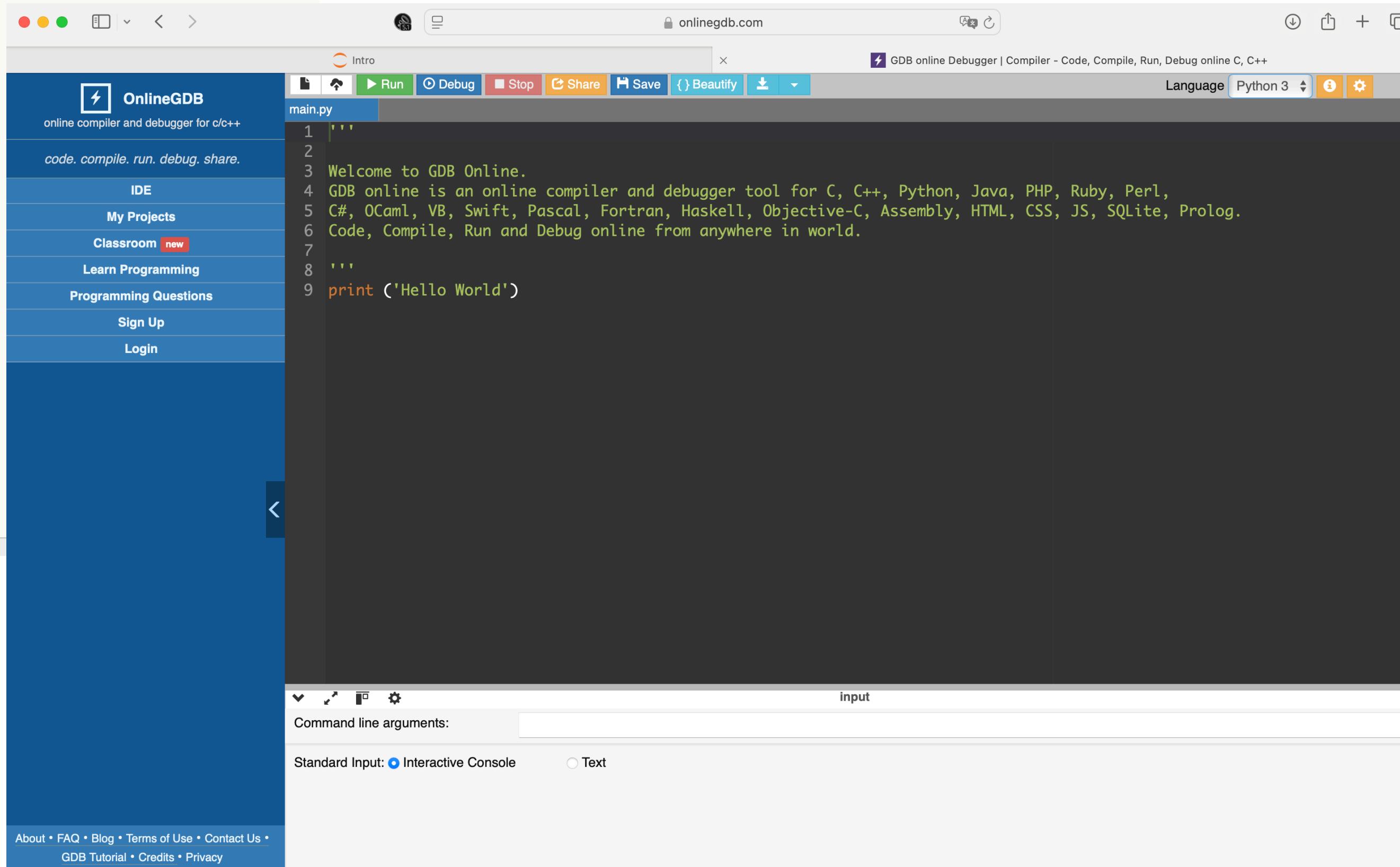
Ale ma mniej funkcji



MOŻECIE W OGÓLE JAKIEŚ PROSTE EDYTORY TYPU MU, ONINEDGB



```
1 class czlowiek:
2
3     # miejsce na konstruktor
4
5     def __init__(klient, imie,wiek,plec):
6         klient.imie = imie
7         klient.wiek = wiek
8         klient.plec = plec
9
10    def przywitaj(klient):
11        print ("Cześć. Mam na imię", klient.imie)
12
13    def ruszaj(klient):
14        if klient.plec == "kobieta":
15            print ("Ruszyłam w drogę")
16        else:
17            print ("Ruszyłem w drogę")
18
```



```
1 """
2
3 Welcome to GDB Online.
4 GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl,
5 C#, OCaml, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite, Prolog.
6 Code, Compile, Run and Debug online from anywhere in world.
7 ...
8
9 print ('Hello World')
```

NAJPOPULARNIEJSZE IDE DO PYTHONA

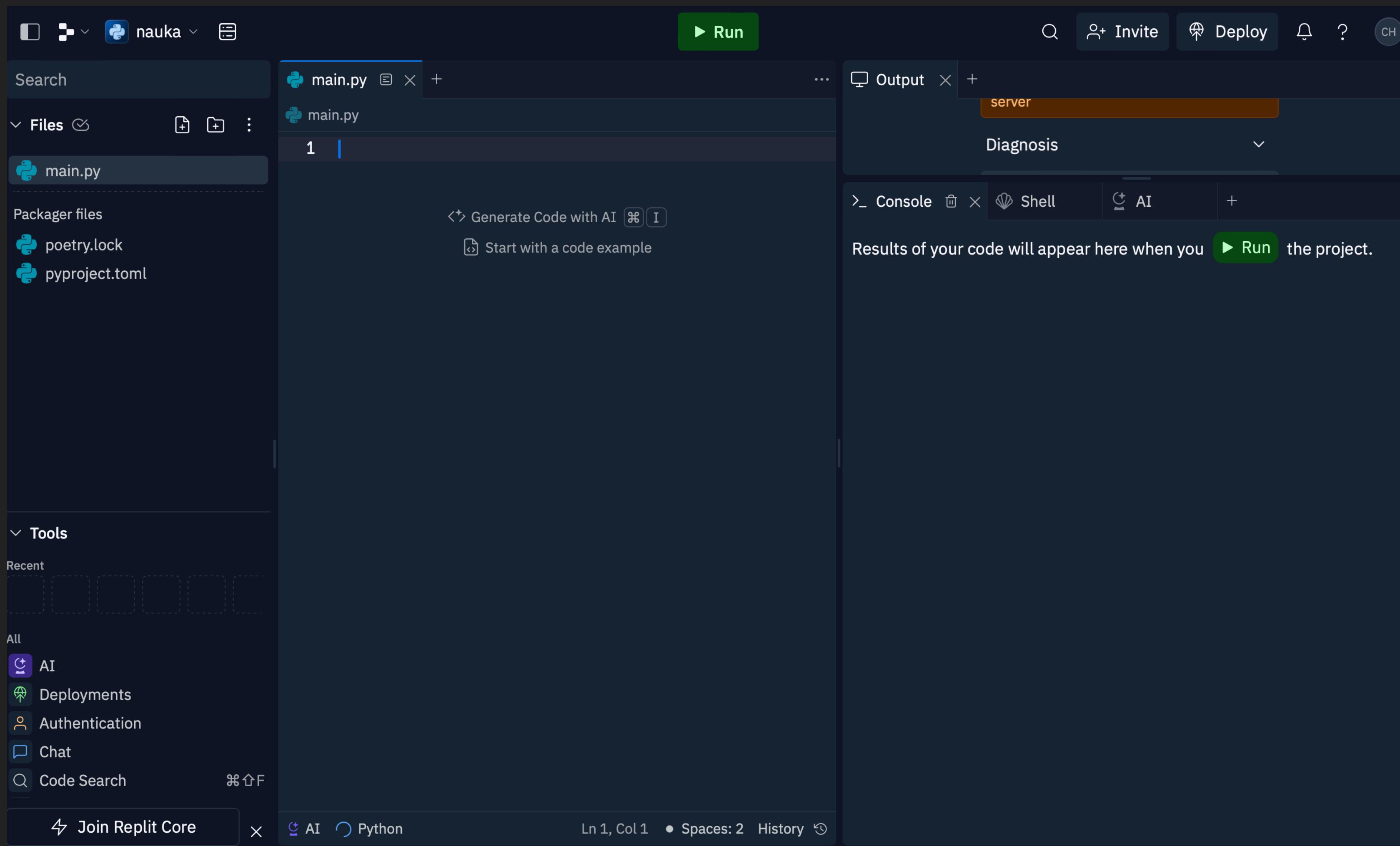
The screenshot shows the Google Colaboratory interface. The top navigation bar includes 'Udostępnij' (Share), a gear icon for settings, and a user profile picture. Below the bar, there are buttons for '+ Kod' (Code), '+ Tekst' (Text), and 'Skopiuj na Dysk' (Copy to Drive). The main content area displays a table of contents on the left and a detailed section on charting in the center. The section on charting includes sub-sections for Matplotlib, Line Plots, Bar Plots, Histograms, Scatter Plots, Stack Plots, Pie Charts, fill_between and alpha, Subplotting using Subplot2grid, Plot styles, and 3D Graphs. A code cell at the bottom shows the import statement: [] `import matplotlib.pyplot as plt`.

GOOGLE COLABORATORY

Fajne, proste i niezależne od platformy.

Ma dużo bibliotek zainstalowanych

NAJPOPULARNIEJSZE IDE DO PYTHONA



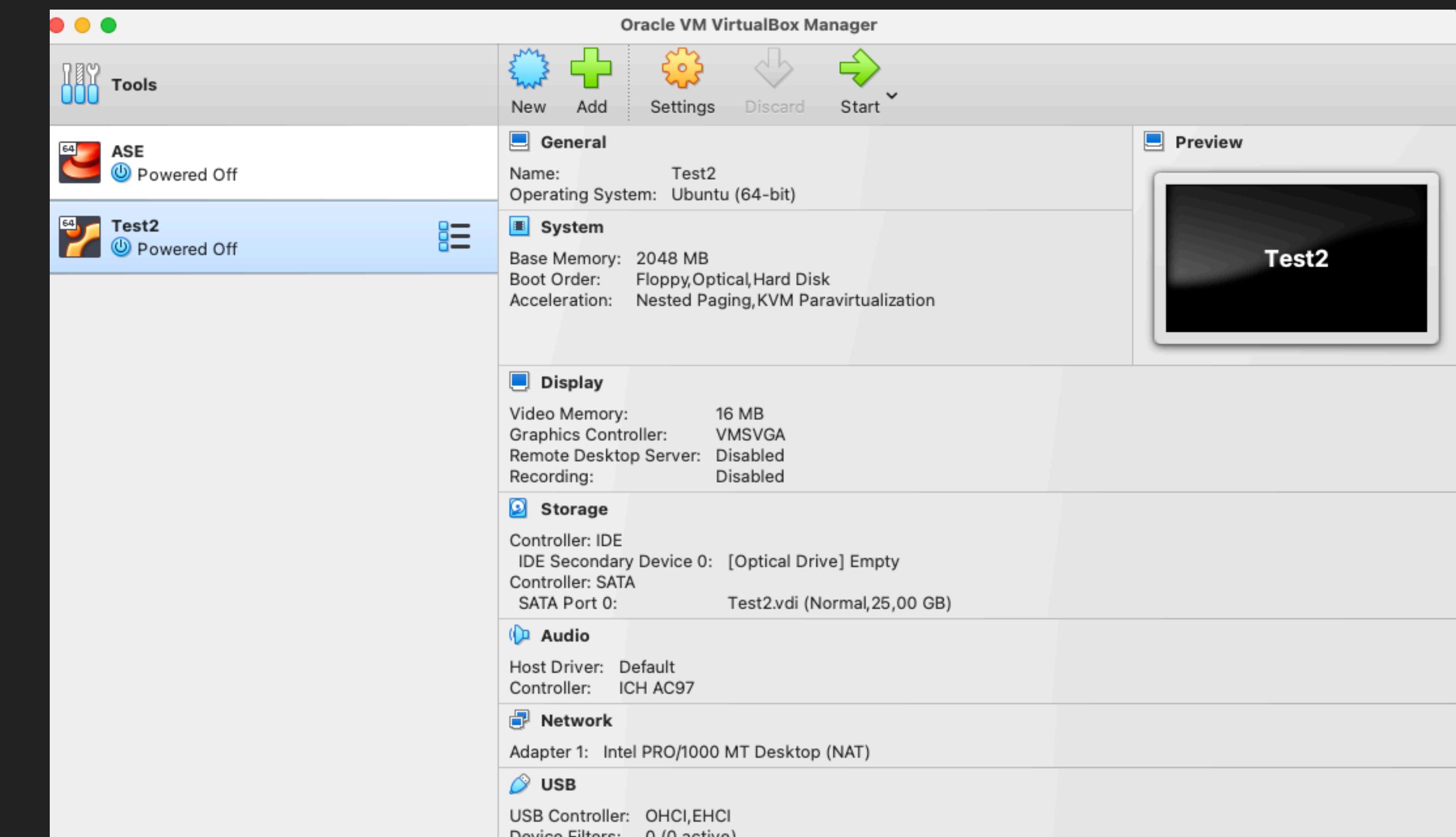
JAK PRACOWAĆ?

ALBO NA SWOIM KOMPUTERZE/
UCZELNIANYM

ALBO WSL, ALBO DOCKER

ALBO POZA COLABEM - JUPYTER

ALBO NA WŁASNEJ MASZYNE WIRTUALNEJ
NP VIRTUAL BOX + LINUX



ZARYS

- ▶ Krótko mówiąc, co to jest Python?
- ▶ Dlaczego Python?
- ▶ Python w porównaniu do innych językach
- ▶ Biblioteki i moduły Pythona
- ▶ Python w akcji

CZYM JEST PYTHON?

- * Językiem programowania bardzo podobnym do PERLa, ale z późniejszymi możliwościami i bardziej zorientowany obiektowo.
- ▶ Użyteczny do tworzenia frameworków sieciowych i innych operacji w internecie.
- ▶ Silne narzędzia przetwarzania zbiorów tekstowych.
- ▶ Dużo użytecznych wbudowanych typów (listy, krotki, słowniki).
- ▶ Jasna przejrzysta struktura języka ogromna sila i ekspresja języka.

DLACZEGO PYTHON?

- * Naturalne narzędzia językowe
- * Łatwy w użyciu interpreter
 - ▶ Wbudowane typy danych dla łańcuchów, list, krotek, słowników,.
 - ▶ Java lub C++ wymagają użycia specjalnych klas w tym celu
 - ▶ Python ma silne możliwości przetwarzania numerycznego – operacje na macierzach
 - ▶ Dobry dla rachunku pdp. i elearningu
 - ▶ Dobry do operacji w Internecie

I SPÓJRZMY NA TEN KOD...

```
x = 34 - 23          # Komentarz  
y = "Hello"          # inny  
z = 3.45  
if z == 3.45 or y == "Hello":  
    x = x + 1  
    y = y + " World"      # łączenie  
print (x)  
print (y)
```

W PRZYPADKU KODOWANIA

- * Podstawienie = a porównanie ==.
- * Wśród liczb używamy +-*/% .
 - ▶ Szczególne użycie + dla łączeniałańcuchów.
 - ▶ Szczególne użycie % dla formatowaniałańcuchów.
- * Logiczne operatory są słowami (**and**, **or**, **not**) **nie symbolami** (&&, ||, !).
- * Podstawową instrukcją drukowania jest “print.”
- * Pierwsze podstawienie pod zmienną tworzy ją.
 - ▶ Zmienne nie muszą być deklarowane.
 - ▶ Python dowiaduje się o typach zmiennych od nich samych.

PODSTAWOWE TYPY DANYCH

* Integer (domyślny dla liczb)

```
z = 5 / 2 # Odpowiedzią jest 2!
```

* Float

```
x = 3.456
```

* String

Mozna używać "" lub " np "abc" 'abc' (to samo)

Jeden łańcuch może się pojawić w drugim "To John's"

Potrójne łańcuchy używamy dla wielolinijkowych łańcuchów """a'b"c"""

INNE TYPY DANYCH

* Long

```
x=1234612974369L
```

* Complex - liczby zespolone

```
x=2.7-3.25e-3j
```

* Listy

```
x=[1, 'a', [3,(3,2-j)], {1:'a'}]
```

* Słowniki

```
x={(1,2,'a'):[3,'s',8.2e-2], 'a':2j}
```

* Krotki

```
x=(2,5,'a',(3,1),[1],{})
```

WCIĘCIA

- * Puste spacje są potężnym narzędziem Pythona: szczególnie wcięcia i nowe linie.
 - ▶ Używaj ENTER kończąc polecenia.
(Nie średnika jak w C++ lub Javie.)
(Używaj \ kiedy musisz przenieść linię.)
 - ▶ Nie używaj { } do oznaczania bloków kodu - to słownik. W zamian używaj wcięć. Pierwsza linia z dodatkowym wcięciem jest traktowana jak początek bloku
 - ▶ Często dwukropki pojawia się jako początek nowego bloku (if :, class :)

KOMENTARZE

- * Komentarz zaczyna się od # pozostałe znaki do końca linii są ignorowane.
- * Można włączyć "docstring" jako pierwszą linię dowolnej funkcji lub klasy.
- * Używaj debuggera i inne narzędzia - używaj je!: w dobrym tonie jest włączyć je do programu.

```
def my_function(x, y):  
    """To jest docstring.  
        Ta funkcja robi ... """  
    # The code would go here...
```

PYTHON I TYPY

Python wyznacza typy danych
automatycznie.

"Dynamiczne Typowanie"

Ale Python nie narzuca tych typów. Ale po zastosowaniu bardzo ich pilnuje.

"Silne Typowanie"

liczba = 42.5
tekst = "To jest przykład"

To spowoduje błąd, ponieważ nie można połączyć int z str bez konwersji
wynik = liczba + tekst

NAZEWNICTWO

* Wielkość liter w nazwach jest istotna. Nazwy nie mogą się zaczynać od cyfry. One mogą zawierać litery, cyfry i znak podkreślenia.

Tomek tomek _tomek __2__tomek__ tomek_2 TomeK

* To są zarezerwowane słowa Pythona:

and, assert, break, class, continue, def, del, elif, else,
except, exec, finally, for, from, global, if, import, in,
is, lambda, not, or, pass, print, raise, return, try,
while

DOSTĘP DO NIE ISTNIEJĄcej NAZWY

- * Dostęp do nazwy, która nie istnieje spowoduje sygnalizację błędu

```
>>> y
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#16>", line 1, in -toplevel-
```

```
    y
```

```
NameError: name 'y' is not defined
```

```
>>> y = 3
```

```
>>> y
```

```
3
```

WIELOKROTNE PRZYPORZĄDKOWANIA

* Można używać wielokrotnych podstawień.

```
>>> x, y = 2, 3
```

```
>>> x
```

```
2
```

```
>>> y
```

```
3
```

```
>>> x, y=y, x
```

OPERACJE NA ŁAŃCUCHACH

- * Można używać wbudowanych metod do operowania zarówno na stałych jak i zmiennych łańcuchowych:

```
>>> "hello".upper()  
'HELLO'
```

- * Jest bardzo dużo dostępnych operacji na łańcuchach.

DRUKOWANIA W PYTHONIE

- * Można wydrukować łańcuch na ekran używając: "print."
- * Używając operatora % w łańcuchu w połączeniu z poleceniem print możemy formatować tekst:.

```
>>> print "%s xyz %d" % ("abc", 34)
```

```
abc xyz 34
```

Print automatycznie dodaje znak nowej linii na końcu łańcucha, chyba, że użyjesz przecinka

```
>>> print "abc" >>> print "abc", "def"
```

```
abc abc def
```

FUNKCJE I DEFINIOWANIE PARAMETRÓW - TU BĘDZIE OSOBNY WYKŁAD NA TEN TEMAT

* `def fun(x, y, z=3.25, l=None):`

* Wywołania

`fun(3,5)`

`t=[2,7,13]`

`fun(*t)`

`sl={'x':5, 'y'=3, 'l'=-7}`

`fun(**t)`

* Dekoratory

`@ gfun`

`def fun(x, y,z=3.25, l=None):`

ANONIMOWE FUNKCJE LAMBDA

To takie funkcje, które można definiować i używać w miejscu, gdzie potrzebujesz krótszej i jednoznacznej funkcji. Można tak kolokwialnie powiedzieć, że takie uproszczone funkcje.

lambda wykaz argumentów:wyrażenie

```
t=[lambda x,y:x+y, lambda x,y:x*y, lambda x,y: x//y]
```

```
>>> t[0](2,3)      5
```

```
>>> t[2](5,2)      2
```

```
>>> print ,%2f' % (lambda x,y: x**y)(2,0.5)  1.414214
```

```
>>> t='32 56 1 7 12 54 ,
```

```
>>> t=t.split()
```

```
>>> t.sort(lambda x,y:cmp(int(x),int(y)))
```

```
>>> t      ['1', ,7', ,12', ,32', ,54', ,56']
```

ELEMENTY PROGRAMOWANIA FUNKCJONALNEGO - TO TAKI PARADYGMAT

`filter(funkcja, sekwencja)`

`map(funkcja, sekwencja)`

`reduce(funkcja, sekwencja)`

* **COMPREHENSIVE LIST**
`e=[(x,y,z) for x in a`

`for y in b if y<"c"]`

`(math.pow(3,y) for y in [2,-1,0,5])`

`[10*i for i in range(10)]`

KLASY I ICH INSTANCJE

```
class VirtualAttributes:  
    __vdict = None  
  
    __vdict_name = locals().keys()[0]  
  
    def __init__(self):  
        self.__dict__[self.__vdict_name] = {}  
  
    def __getattr__(self, name):  
        return self.__vdict[name]  
  
    def __setattr__(self, name, value):  
        self.__vdict[name] = value  
  
class A:  
    def __init__(self):  
        self.__X = 3  
  
class B(A):  
    def __init__(self) :  
        A.__init__(self)  
        self. X = 37
```

ZASTOSOWANIA

Obszar	Narzędzia i rozszerzenia
Systemowe programowanie: podtrzymywane dla narzędzi wszystkich poziomów	Gniazda sieciowe, procesy, wielowątkowość, sygnały, RPC, katalogi, POSIX
GUI: różne narzędzia GUI	Tkinter, wxPython, PyQt, PyGTK, Anygui, Swing, PythonCard, Dabo ...
Interface bazodanowy: interface dla zarówno relacyjnych jak i obiektowych baz danych	MySQL, Oracle, Sybase, PostgreSQL, SQLite, ZODB, DBM, ...
Narzędzia Windows: dostęp do różnych narzędzi Windows	MFC wrapper, interface COM, skrypty ActiveX, ASP, Drivery Odbc, .NET, ...
Narzędzia sieciowe: gniazda, CGI, narzędzia klienta i serwera, framework sieciowy, parsery, podtrzymywanie Apache, integracja z Java	JYTHON, XML, email, ElementTree, htmllib, telnetlib, urllib, Zone, CherryPy, Twisted, Webware, Django, mod_python, SSL ...
Dystrybucja obiektów: SOAP web serwisy, XML-RPC, CORBA, DCOM	PySOAP, SOAPy, xmlrpclib, ILU, Fnorb, omniORB, PyWin32 ...
Inne popularne narzędzia: grafika, języki wizualizacji, matematyka, kryptografia, integracja, gry,	PIL, VPYTHON, Blender, PyOpenGL, NLTK, YAPPS, VTK, NumPy, PyCrypto, SWIG, ctypes, PyGame, MoinMoin ...

Moduły os i glob

`os.popen(polecenie[, tryb[, rozmiar_bufora]])`

`os.walk(top[, topdown=True [, onerror=None]])`

`glob.glob('[acu]os*.ht??')`



Moduły pickle, cpickle, tarfile i zipfile

```
pickle.dump(x, f)  
x = pickle.load(f)
```

```
fileTar=open(filename [,mode [,fileobj [,buffersize]]])  
    fileTar.add(name [, arcname [,recursive]])  
fileTar.extract(file [,path])
```

```
zipfile(filename [, mode [, compression]])  
write(filename [, arcname [,compreression]])  
read(filename)
```



Moduły `thread` i `threading` i wielowątkowość

```
import thread, time
def print_time(threadName, delay):
    while 1:
        time.sleep(delay)
        print "%s: %s" % (threadName,
                           time.ctime(time.time()))
    thread.start_new_thread(print_time, ("Wątek 1",2,))
    thread.start_new_thread(print_time, ("Wątek 2",4,))
    while 1:
        pass
```

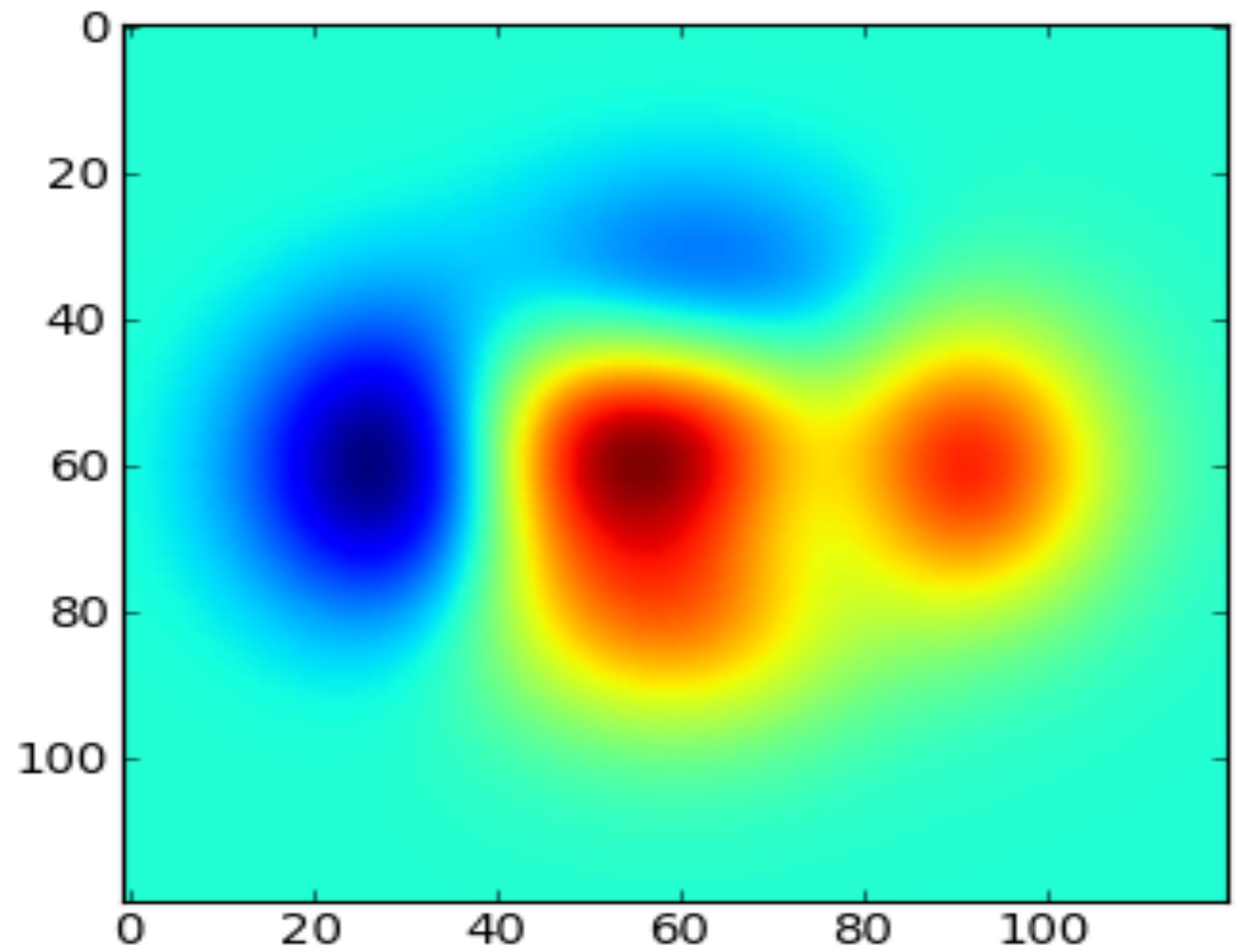


Moduł urllib – pobieranie adresów url

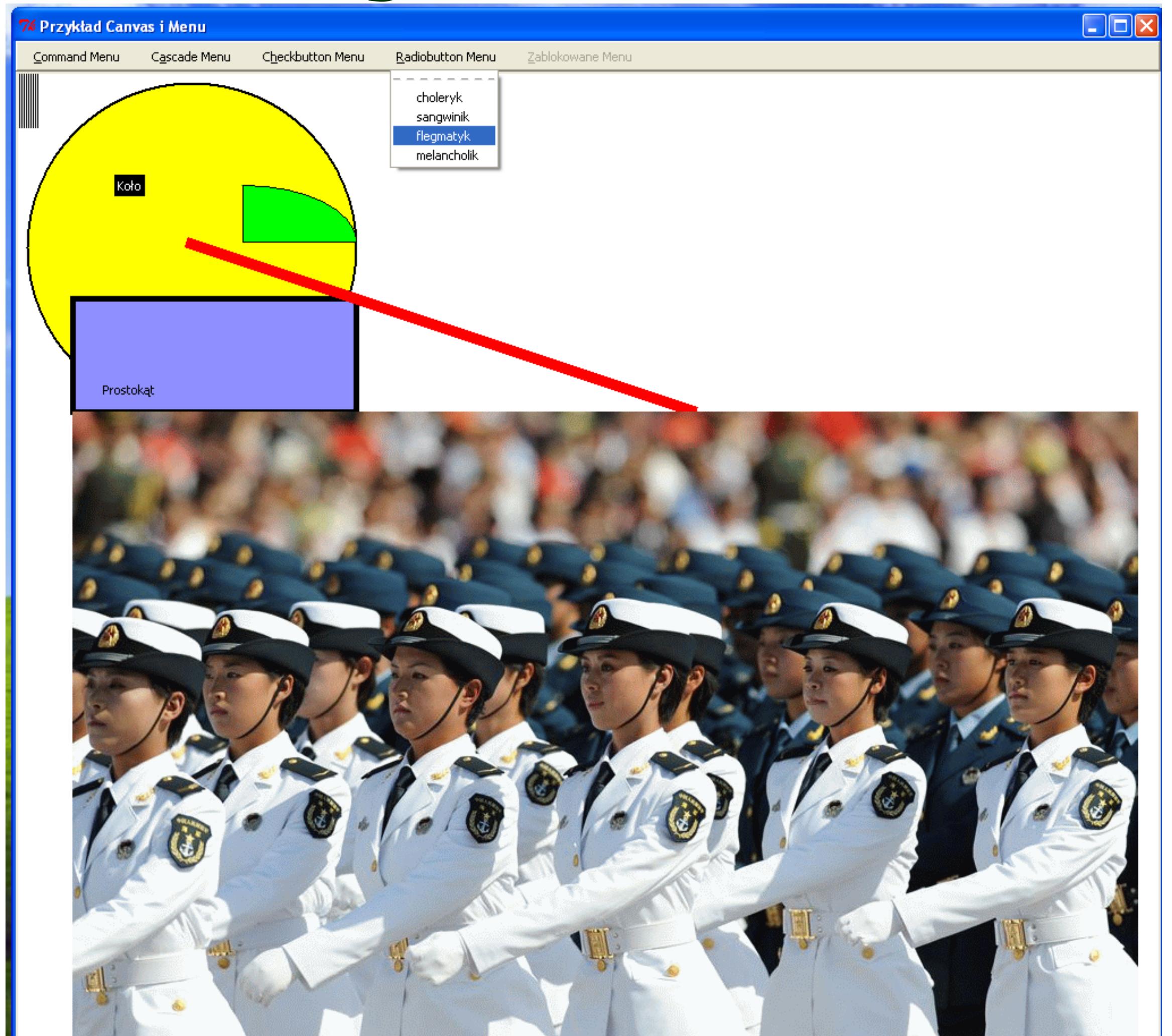
```
u=urllib.urlopen(url)  
buffer=u.read()  
u.info()  
u.geturl()
```



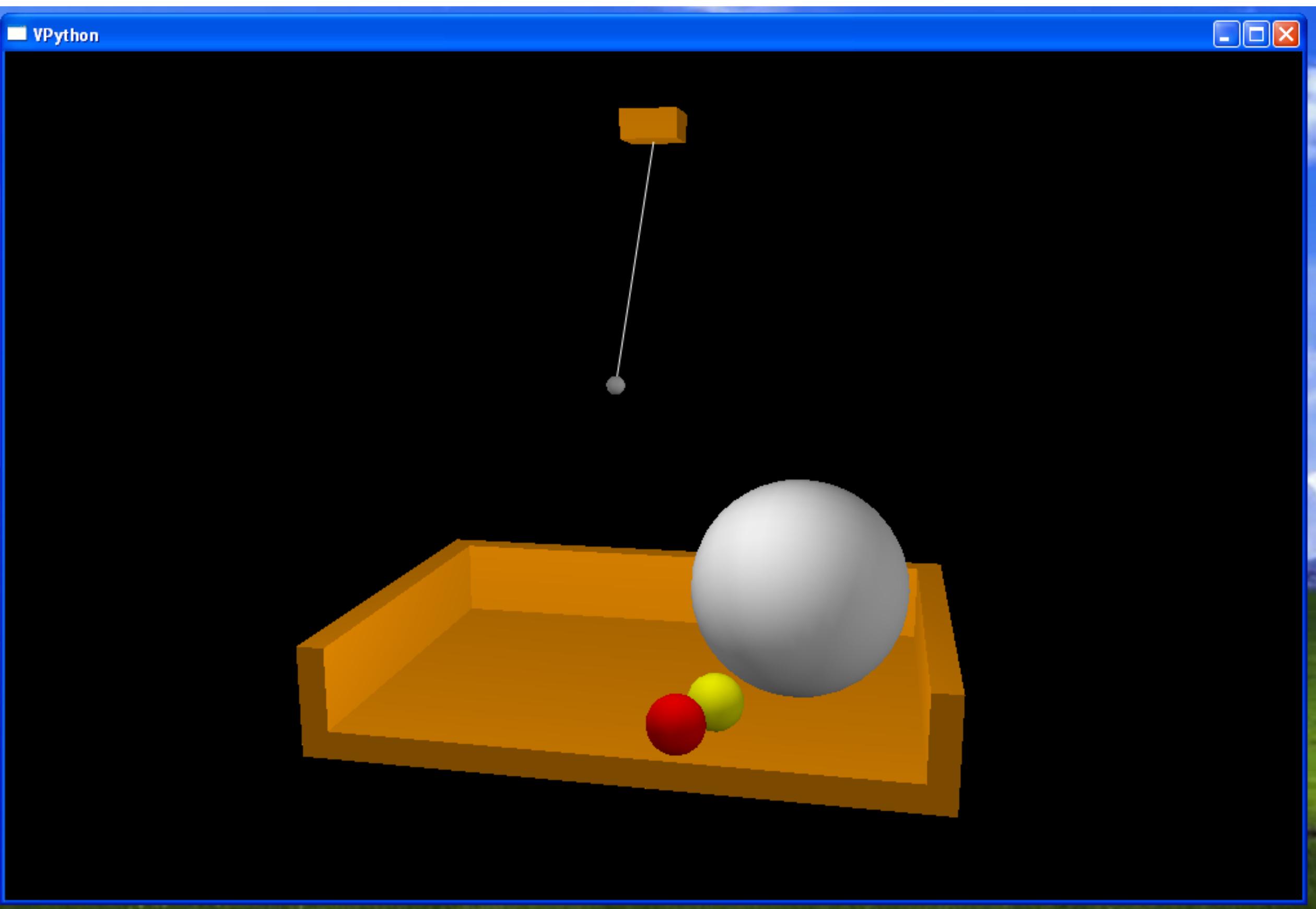
Matplotlib i wykresy



Tkinter i interfejs użytkownika



Vpython i wizualizacja



DJANGO i model MVC

model – widok - kontroler

- `models.py` - opis tabel baz danych reprezentowanej przez klasę w języku Python. Klasę tę nazywamy modelem. Używając jej, możesz tworzyć, pobierać, aktualizować i usuwać rekordy w bazie danych za pomocą prostego kodu w Pythonie bez potrzeby sięgania cały czas po bardzo podobne polecenia SQL.
- `views.py` zawiera logikę biznesową strony. Funkcja `latest_books()` nosi nazwę widoku.
- `urls.py` określa, który widok zostanie wywołany dla wskazanego adresu URL.
- `latest_books.html` to szablon HTML opisujący wygląd strony WWW. Wykorzystuje język szablonowy z prostymi poleceniami logicznymi, np. `{%for book in`